A Novel Class of Multi-Agent Algorithms for Highly Dynamic Transport Planning Inspired by Honey Bee Behavior


Abstract — Commercial transport planning as well as individual intra-city or inter-city traffic in densely populated regions, both in Europe and the US, increasingly suffer from congestion problems, to an extent which e.g. affects predictable transport planning substantially (except – so far – for overnight tours). Due to the highly dynamic character of congestion forming and dissolving, no static approach like shortest path finding, applied globally or individually in car navigators, is adequate here: Its use even makes things worse as can be frequently observed. In this paper we present a completely decentralized multi-agent approach (termed BeeJamA) on multiple layers where car or truck routing are handled through algorithms adapted from the BeeHive algorithms which in turn have been derived from honey bee behavior. We report on extensive distributed simulation experiments in the BeeJamA project which demonstrate a very substantial improvement over traditional congestion handling.

I. INTRODUCTION

Traffic Congestions and Transport Planning. In densely populated European regions like Holland or Germany even wide-area transport planning easily comes to its limits, due to rapidly increasing congestion problems, not only on inner-city roads but also on national routes or interstate freeways. Except for weekends, with its heavy restrictions on commercial traffic German radio stations broadcast congestion reports on freeways every 30 minutes but only for traffic jams that exceed 3 (sometimes 5) km in length. (It would take too much time reporting the shorter ones.) The increasing geographic density of congestive situations creates several serious and complex problems regarding the timely arrival of goods or persons, i.e. of minimizing transportation time and distances: As transport companies utilize central planning and guidance procedures individual car drivers or truckers may in turn rely on built-in navigators yet these execute the same algorithms for computing “shortest” detour paths. Either way: As a typical result even much heavier congestions occur, due to the highly dynamic behavior of the whole system.

In this paper we introduce a novel, highly adaptive algorithm for on-line dealing with, if not avoiding, congestions before they occur. We have borrowed the main ideas from Swarm Intelligence as they have been detected in the honeybee communication [1,2]. Over the past 5 years, in the BeeHive project, we have developed distributed multi-agent algorithms for network routing [4,6,7,8,9] which have been found to be considerably superior in the field, regarding e.g. flexibility, real-time, fault tolerance. In the spirit of this approach we will define, in this paper, a tailored multi-layer distributed routing algorithm termed BeeJamA (ideal- ly aimed at traffic jam avoidance). We will demonstrate its quality w.r.t. both timely reactions to upcoming congestions and finding appropriate detours. This is a key step towards a realistic transport planning under ever present congestions. While right now there is not yet any time guarantee for detour paths – a consequence of the highly dynamic nature of the problem – we at least have QoS results regarding minimizing transportation times.

Previous and Related Work. Our earlier theme-related work has been quoted in the previous paragraph. Other than that there has been, for the past few years, an enormous amount of work and generous funding for traffic control, e.g. transport planning [18], quite a novel prediction mechanism for traffic jams (although restricted to freeways) [10,11] based on broadcasting of congestion information. A broad coalition of car manufacturers and public institutions, both in Europe and the US, have advertised automotive-related research and development covering both crash and congestion avoidance [17]. While in [17] congestion avoidance has not even been addressed so far the work in [18] relies on a decentralized form of static detour planning based on jam information. Detours in [17], if not prescribed statically, are computed individually through static algorithms. To the best of our knowledge our own approach is the first one to explicitly deal with the avoidance of traffic jams.

Organization of the Paper. In section II we will briefly introduce the original BeeHive algorithm for adaptive routing in packet switching networks. Section III is devoted to introducing the BeeJamA model and algorithm with a basic street model. Based on real data for congestive behavior we define, in section IV, a mathematical quality rating function needed for directive decisions in the routing
process. Extensive comparative simulations on a realistic street model reported in section V reveal the very remarkable advantage of BeeJamA over standard routing algorithms as used in automated navigators. The last section summarizes the results and discusses an outline for our ongoing and future work.

II. BEE INSPIRED ROUTING

A. Bees in Nature

A honey bee colony manages to react to countless changes in the forage pattern outside the hive, and to internal changes inside the hive, through a decentralized and sophisticated communication and control system. A honey bee colony can thoroughly monitor a vast region around the hive for rich food sources, nimbly redistribute its foragers within an afternoon, fine-tune its nectar processing to match its nectar collecting, effect cross inhibition between different forager groups to boost its response differential between food sources, precisely regulate its pollen intake in relation to its ratio of internal supply and demand, and limit the expensive process of comb building to times of critical need for additional storage space [1]. A bee colony demonstrates this flexible and adaptive response because it is organized with morphologically uniform individuals yet working in different roles, under temporary specializations. A bee takes up four roles during her lifetime: cleaner, nurse, foodstorer and forager. The foragers could be further recognized as nectar collectors, pollen collectors and water collectors [1]. The foragers take up two type of functional roles within each subspecialty: scouts, which discover new food sources around the hive, and foragers, which transport nectar from an already discovered flower site by following the dances of other scouts or foragers.

In 1944, Nobel Laureate Karl von Frisch reported in his book “Tanzsprache und Orientierung der Bienen” [2] (translation done by Chadwick [3]) how the foragers use two type of dances: round dances, which show that a food source is present in the near vicinity of the hive (within about 100 meters), and waggle dances which further specify the direction and distance to a distant food source (up to a few kilometers). In total the recruited foragers arrive in greater numbers at more profitable food sources because the dances for richer sources are more conspicuous and hence likely to be encountered by the unemployed (dance-following) foragers [1].

B. The BeeHive Algorithm

In our initial work we modeled bee agents in packet switching networks. For the purpose of finding suitable paths between sites, we extensively borrowed from the principles behind bee communication. Through this work we developed novel network routing protocols BeeHive and BeeAdHoc (for wireless ad-hoc communication) that proved far superior to common routing protocols, both single and multipath (e.g. OSPF, DGA, etc.) [6, 7]. In the following we describe our BeeHive algorithm. In section IV we adapt this algorithm to solve our vehicle routing problem.

As mentioned before honey bees evaluate the quality of each discovered food site and only perform the waggle dance if the quality is above a certain threshold. Thus, not each discovered site receives attention. As a result, quality flower sites are exploited quite extensively. We abstract a dance floor into a routing table where bee agents, launched from the same source but arriving from different neighbors at a given node, could exchange routing information for modeling the network state at this node.

The majority of foragers are found to exploit the food sources in the closer vicinity of their hive while a minority among them visits food sites far away from their hive. We transformed this observation into an agent model that has two types of agents: short distance bee agents and long distance bee agents. Short distance bee agents collect and disseminate routing information in the neighborhood (up to a specific number of hops) of their source node while long distance bee agents collect and disseminate routing information to typically all nodes of a network. Informally, the BeeHive algorithm and its main characteristics can be summarized as follows:

1. The network is organized into fixed partitions called foraging regions. A partition results from particularities of the network topology. Each foraging region has one representative node. If this node crashes then the next higher IP address node takes over the job.
2. Each node also has a node-specific foraging zone which consists of all nodes from which short distance bee agents can reach this node.
3. Each non-representative node periodically sends a short distance bee agent, by broadcasting replicas of it to each neighbor site.
4. When a replica of a particular bee agent arrives at a site it updates routing information there, and the replica will be flooded again, however, it will not be sent to the neighbor from where it arrived. This process continues until the life time (number of hops) of the agent has expired, or if a replica of this bee agent had been received already at a site. In the latter case the new replica will be killed there.
5. Only representative nodes launch long distance bee agents that would be received by the neighbors and propagated as in 4. However, their life time (number of hops) has a higher limit, the long distance limit.
6. The idea is that each agent while traveling, collects and carries path information, and that it leaves, at each node visited, the trip time estimate for reaching its source node from this node over the incoming link. Bee agents use priority queues for quick dissemination of routing information.
7. Thus each node maintains current routing information for reaching nodes within its foraging zone and for reaching the representative nodes of foraging regions. This mechanism enables a node to route a data packet (whose destination is beyond the foraging zone of the given node) along a path toward the representative node of the foraging region containing the destination node.

8. The next hop for a data packet is selected in a probabilistic manner according to the quality measures assigned to the current node’s edges to its neighbors. As a result, not all packets follow “best” paths. This will help in maximizing the system performance though a data packet may not follow a best path, a concept directly borrowed from a principle of bee behavior: A bee could only maximize her colony’s profit if she refrains from broadly monitoring the dance floor to identify the single most desirable food [1] (in comparison OSPF always chooses a next hop on the shortest path).

Figure 1 provides an exemplary partitioning of the flooding algorithm. Short distance bee agents can travel up to 3 hops in this example. Each replica of the bee agent launched by Node 10 is specified with a different trail to identify its unique path. The numbers on the paths show their quality (costs). The flooding algorithm is a variant of the breadth first search algorithm. Nodes 2,3,4,5,6,7,8,9,11 constitute the foraging zone of node 10.

![Figure 1: BeeHive Flooding Algorithm](image)

For routing in packet switching networks the quality of an edge (costs in fig. 1) is approximated by the trip time that a packet will take if sent over that edge. In our estimation model bee agents approximate the trip time \( t_{ij} \) that a packet will take for reaching their source node \( s \) from current node \( i \) (ignoring the protocol processing delays for a packet at node \( i \) and \( s \)) as follows:

\[
t_{ij} = \frac{ql_{in}}{b_n} + tx_{in} + pd_{in} + t_{in}
\]  

(1)

where \( ql_{in} \) is the size of the queue (in bits) for neighbor \( n \) at node \( i \), \( b_n \) is the bandwidth of the link between node \( i \) and neighbor \( n \), \( tx_{in} \) and \( pd_{in} \) are transmission delay and propagation delay respectively of the link between node \( i \) and neighbor \( n \), and \( t_{in} \) is trip time from \( n \) to \( s \). Bandwidth and propagation delays of all links of a node are approximated by transmitting hello packages. (For more details see e.g. [4,9]).

III. THE BEEJAMA ALGORITHM

The algorithm presented in this section is designed for routing vehicles on roads and freeways, with the intention of avoiding traffic congestions. It is based on the BeeHive algorithm with a few distinct changes to adapt to the problem of vehicle routing instead of packet switching. The modified algorithm is called BeeJamA.

For the BeeHive Algorithm to adapt to the highly dynamic problem of routing vehicles, and to avoid traffic congestions, we followed a layered approach where cars are routed from intersection to intersection on a next hop basis. On the first layer edges represent roads and nodes represent intersections. We call this layer the area layer (see fig. 2). Different from packet switching networks, traffic intersections do not possess the capability to maintain routing tables and communicate with approaching or crossing cars. Thus their task is taken over by a single responsible navigator for each area. This local navigator manages the routing tables for the nodes in its area and maintains communication with each vehicle in its area as well. The area size of a single navigator will be designed to be large enough to offer sufficient alternative routes to cope with major traffic incidents (e.g. blockage of a highway lane) but small enough to allow timely next-hop-selections before the next road intersection is reached.

![Figure 2: Layered Routing Model](image)

Due to the lack of vehicular hello packages (BeeHive utilizes hello packages to determine the quality of its edges) all cars continuously transmit their position, speed and destination to their responsible navigator. The navigator uses this information to update the information in its routing tables and to supply vehicles with appropriate routing information in time before reaching the next node.
In BeeHive hop qualities are calculated from propagation and queueing delays but the quality of a road has to be estimated in terms of different attributes. In section IV we introduce the traffic quality functions utilized by BeeJamA. For the time being we assume that each road is given a quality that reflects its length, its allowed maximum speed, the vehicle density, etc. (We will still use the term costs.)

Routing across several areas is managed on the net layer. On this layer nodes represent areas and edges represent roads that connect neighboring areas. If more than one road connects two neighboring areas, the edge on the net layer is rated with the single highest quality (lowest travelling costs) of those roads (Please note that we route single vehicles as we cannot utilize more than one route at a time like in packet switching networks).

For the sake of conceptional clarity the following routing mechanisms are demonstrated with a very simple “honeycomb” model. Each area consists of 7 nodes and each area has exactly 6 neighboring areas to each of which it is connected by a single road (with one lane in each direction). This basic “honeycomb” model is depicted in figures 3 and 4.

A. Routing across the net layer

Usually automobiles must cross several areas to reach their individual destinations. For routing across the net-layer the network is partitioned into fixed foraging regions, and each node maintains a specific foraging zone that consists of all neighboring nodes within a certain hop range (this is identical to the standard BeeHive procedure, see III.B). Figure 3 depicts two foraging regions (A, E, I, M, H, L and B, C, D, F, G, J, K, N) and the foraging zone of node (area) A (B, C, E, F, H, I).

![Network Layer](image)

For routing on the net layer three types of routing tables are needed: The Inter Foraging Region table (IFRnet), Intra Foraging Zone table (IFZnet) and the Foraging Region Membership table (FRMnet). The updating process for all routing tables is similar to BeeHive and done by software bee agents. But instead of recording propagation and queueing delays to calculate routing costs for updating trip costs, bee agents propagate the (known) qualities of the corresponding roads (their latest information, see the example in section V). The next area on a vehicle’s route to its destination is selected probabilistically according to the costs of the routes in the current node’s routing tables (see II.8).

The IFZnet table stores routing information for all the nodes in its foraging zone. The table contains the costs of all routes to a node within its foraging zone by travelling over a direct neighbor (see table 1). Thus, the IFZnet table of a node A has a size of $O(N_d Z_d)$, where $N_d$ is the number of direct neighbor nodes of node A and $Z_d$ is the number of all nodes in the foraging zone of node A. Table 1 depicts the IFZnet table of node A.

<table>
<thead>
<tr>
<th>IFZnet</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>F</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$c_{NB}$</td>
<td>$c_{NC}$</td>
<td>$c_{NE}$</td>
<td>$c_{NF}$</td>
<td>$c_{NH}$</td>
<td>$c_{NI}$</td>
</tr>
<tr>
<td>E</td>
<td>$c_{EB}$</td>
<td>$c_{EC}$</td>
<td>$c_{EE}$</td>
<td>$c_{EF}$</td>
<td>$c_{EH}$</td>
<td>$c_{EI}$</td>
</tr>
</tbody>
</table>

Table 1: Intra Foraging Zone Table of Node A

Hence, $c_{NC}$ represents the costs of traveling from node A to node C over node B.

The FRMnet table maps each node to its foraging region and thus consists of two rows equal in size, to the total number of nodes on the net layer (see table 2).

<table>
<thead>
<tr>
<th>FRMnet</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_A</td>
<td>R_B</td>
<td>R_C</td>
<td>R_D</td>
<td>R_E</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Foraging Region Membership Table

In our example node C belongs to region R_K, which has node K as its representative node.

The IFRnet table (see table 3) stores routing information to representative nodes in the network. If a node has to be reached that is not known in the IFZnet table of the current node, IFRnet provides routing information to the representative node of the destination node’s foraging region (which is known via the FRMnet table).

<table>
<thead>
<tr>
<th>IFRnet</th>
<th>R_A</th>
<th>R_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$c_{NB}$</td>
<td>$c_{NC}$</td>
</tr>
<tr>
<td>E</td>
<td>$c_{EB}$</td>
<td>$c_{EC}$</td>
</tr>
</tbody>
</table>

Table 3: Inter Foraging Region Table of Node A

Hence, the IFRnet table of a node A has a size of $O(N_d R)$, where $N_d$ is the number of direct neighbor nodes of node A and $R$ is the number of all representative nodes on the net-layer. In our example with the two representative nodes A and K, $c_{KE}$ represents the costs for travelling towards region $R_K$ over neighbor E.

B. Routing on the area layer

Once the next area for a vehicle’s route to its destination is selected on the net layer, the vehicle has to be routed on the area layer. Vehicles either want to reach a destination within the current area or cross the area to reach a destination within a different area. Areas are connected by so called border nodes. Border nodes have at least one edge in common with a node from a different area. In our basic
example (see fig. 4) each area has 6 border nodes, which connect one area with each of its 6 neighbors.

For routing on the area layer a major adjustment is made to the network partitioning of standard BeeHive:

*There is only one foraging zone for all nodes within an area, and the foraging zone coincides with the area.*

Since the layout of an area does not change over time (except after extensive street construction works) and all nodes within one area are managed by a single navigator, every node knows the routes to all nodes within its area. In our example the common foraging zone/region for each node within area A consist of all nodes 1, 2, 3, 4, 5, 6, 7 (see figure 4).

![Fig. 4: Area Layer in the Basic “Honeycomb” Model](image)

Routing itself is again based on the standard BeeHive algorithm and utilizes the three types of routing tables: The *Inter Foraging Region* table (IFR\(_{area}\)), *Intra Foraging Zone* table (IFZ\(_{area}\)) and the *Foraging Region Membership* table (FRM\(_{area}\)).

The IFZ\(_{area}\) table is similar to the IFZ\(_{net}\) table and contains information about the costs to reach each node in its foraging zone. Thus, the IFZ\(_{area}\) table of a node \(x\) has a size of \(O(N_xD_x)\), where \(N_x\) is the number of neighbors of node \(x\) and \(D_x\) is the number of nodes within the area of \(x\). Table 4 depicts the IFZ\(_{area}\) table of node 4.

![Table 4: Intra Foraging Zone Table of Node 4](image)

Hence, \(c_{71}\) represents the costs of traveling from node 4 to node 7 over node 1.

On the area layer the IFR\(_{area}\) table is significantly different from the IFR\(_{net}\) table. The IFR\(_{area}\) area table consists of routing information about the transitions to other areas over border nodes. Thus, the IFR\(_{area}\) table is unique for each area, and equal for all nodes within an area.

Please note that the table is sparsely populated. Although costs for transitions to a neighboring area could be calculated for all border nodes (e.g. the costs to reach area B over node 7 and thus over area E), entries are only non-empty for a transition from a border node \(x\) to an area \(Y\) if node \(x\) has at least one edge in common with a node from area \(Y\). This is done because the next-area selection is done on the net layer, and it should not eventually be in conflict with selections on the area layer. Table 5 depicts the IFR\(_{area}\) table for area A.

![Table 5: Inter Foraging Region Table of Area A](image)

The size of the IFR\(_{area}\) table of area A is \(O(BN_A\cdot N_A)\), where \(BN_A\) is the number of border nodes in area A, and \(N_A\) is the number of neighboring areas to area A.

The FRM\(_{area}\) table is a mapping of nodes to areas (and thus is most suitable described as a road directory). The size of the FRM\(_{area}\) table is \(2N\), with \(N\) being the total number of nodes in the network. Table 6 depicts the FRM\(_{area}\) table of our basic example network.

![Table 6: Foraging Region Membership Table of the Network](image)

**IV. QUALITY RATING FUNCTION**

One of the key challenges of applying the BeeHive algorithm to vehicle routing is the development of an appropriate rating function for road traveling costs. The rating function is utilized to evaluate each edge on the area layer reflecting the actual traffic situation on the corresponding road section. In BeeHive edge qualities are calculated via propagation and queuing delays to enhance throughput in packet switching. BeeAdHoc utilizes measured energy consumption to rate different routes in MANETs with the objective of maximizing battery lifetimes.

In *BeeJama* the focal point is to avoid traffic congestions but the avoidance of heavy traffic may be adverse to the objectives of an individual driver. Long detours that might lower the traffic density on a stressed road may not be acceptable to individual drivers and result in overruling or ignoring the system’s routing recommendations by a majority of the drivers. Thus a rating function will be designed to cope with the individual needs for the fastest possible route, and at the same time follow the system’s objective to avoid traffic congestions.

The leading cause for traffic congestions is an excessive density of vehicles on a road that leads, in combination with unpredictable acceleration and breaking behavior of individual drivers, to an average travelling speed that is far below the maximum travel speed of that particular road.

In order to avoid traffic congestions, an intuitive approach is to reduce the vehicle density on already crowded
roads by recommending detours to vehicles involved, away from that particular road.

Through extensive empirical studies in [5, 16] the interdependencies of average vehicle speed, a street’s vehicle density and the traffic quality (in terms of congestion) have been studied for 2-lane highways, 4-lane and 6-lane freeways [14, 15]. Figure 5 shows the progression of the approximated density-speed-curve for 4-lane freeways.

![Density-Speed Curve for 4-Lane Freeways](image)

**Fig. 5:** Calculated p-v-Diagram for 4-Lane Freeways, taken from [5]

The curve is partitioned into 3 sections characterizing a congestion free state, a transitional state rather difficult to interpret, and traffic congestions. (Due to space limitations we restrict ourselves to the discussion of 4-lane freeways at this point.) The p-v-curves for highways and 6-lane freeways are of similar shape but differing in their maximum speed \(v_{\text{max}}\) (intersection with the v-axis), and in the points at which the three different traffic states merge [5].

For our model we approximated the p-v-curves with three functions over three separate intervals. For \(0 < \rho \leq \alpha\), \(\alpha < \rho \leq \beta\) and \(\rho > \beta\) where \(\alpha\) and \(\beta\) specify the vehicle densities at which the transitions between the three different states can be observed. Two parameters \(A\) and \(B\) are used to specify the average speeds at which the three states merge.

The functions are:

\[
\begin{align*}
\rho &< \alpha: & v_{\text{edge}} &= \left(\frac{A - v_{\text{max}}}{\alpha\rho} + v_{\text{max}}\right) \quad (2) \\
\alpha &< \rho \leq \beta: & v_{\text{edge}} &= \left(\frac{\rho - \alpha(B - A)}{\beta - \alpha} + A\right) \quad (3) \\
\rho &> \beta: & v_{\text{edge}} &= \frac{\beta^{\rho}B}{\rho} \quad (4)
\end{align*}
\]

The parameters for the different road types are given in table 7, and the calculated p-v-curves are found in figure 6.

<table>
<thead>
<tr>
<th>Road Type</th>
<th>(v_{\text{max}}) [km/h]</th>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(A)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway</td>
<td>100</td>
<td>35</td>
<td>40</td>
<td>72</td>
<td>20</td>
</tr>
<tr>
<td>4-lane freeway</td>
<td>120</td>
<td>45</td>
<td>55</td>
<td>86</td>
<td>40</td>
</tr>
<tr>
<td>6-lane freeway</td>
<td>120</td>
<td>50</td>
<td>70</td>
<td>86</td>
<td>40</td>
</tr>
</tbody>
</table>

**Table 7:** Parameters for Different Road Types

The costs of an edge would then be calculated as:

\[
c_{\text{edge}} = \frac{l_{\text{edge}}}{v_{\text{edge}}} \quad (5)
\]

where \(l_{\text{edge}}\) is the length of the relevant edge. Thus, the costs for travelling over an edge may be interpreted as the estimated travel time over that edge.

In our estimated model the trip costs \(c_{i}^{\text{in}}\) for a vehicle to travel from its current node \(i\) to its destination node \(s\) are then calculated, in analogy to BeeHive (see II.B) according to:

\[
c_{i}^{\text{in}} \approx c_{i}^{\text{in}} + c_{in} \quad (6)
\]

Here \(c_{i}^{\text{in}}\) is the cost for travelling from the current node \(i\) to a neighboring node \(n\), and \(c_{in}\) is the cost for travelling from that neighbor to the destination node \(s\).

Bees in a bee hive perform the waggle dance only if the quality of a discovered food source exceeds a certain threshold. Otherwise, the food source is discarded and no new foragers will be recruited. In analogy to this behavior, a route \(r\) may only be selected if its travel cost \(c_{r}\) does not exceed a dynamical threshold \(t_{0} \geq c_{r}\) (and thus would not result in detours unacceptable to individual drivers).

V. SIMULATION STUDIES

In the course of our project work we developed a traffic simulator to satisfy our needs to test and evaluate BeeJamA in different scenarios against common routing algorithms. To emulate realistic traffic movement we utilized well-established traffic models developed by Nagel and Schreckenberg [10, 11]. They introduced a stochastic discrete automaton model to simulate freeway traffic. We utilized advancements to this model made by M. Rickert [12], introducing a rule set to allow passing maneuvers on multilane roads, as well as improvements made by W. Knospe [13] incorporating anticipation effects, reduced acceleration capabilities, and an enhanced interaction horizon for breaking. Cellular automata are step-based, and sequentially compute each cell’s next state based on a set of probabilistic rules, depending on the current cell’s state. More details about our traffic simulator would carry us beyond the page limitations.

We set up several simple scenarios (e.g. the basic “honeycomb” model) as well as realistic road scenarios based on commercially available topological road data of the
eastern German Ruhr District (see figure 7). We evaluated BeeJamA against Dijkstra-based fastest (in terms of travel times) path routing which is utilized in most of today’s car navigational systems in combination with regularly updated traffic information. In most European countries traffic message channel broadcasting (TMC) is utilized to supply navigational systems with up-to-date traffic information. TMC updates are usually made available every 5-20 minutes. For commercial reasons TMC updates often are further delayed by up to 20 minutes. In order to compare BeeJamA with a more ideal conventional system, we assume that accurate data updates are made available to Dijkstra routed vehicles every 10 minutes. In our simulations we evaluate both, the ability to avoid system wide traffic congestions as well as individual travel times to destinations.

We conducted a series of comparative experiments with a section of the Ruhr District and traffic generated on 4 nodes. All drivers try to reach one common destination.

Experiments were conducted 10 times each, for BeeJamA based routing and Dijkstra routing. We monitored each road’s travel speed throughout the experiments.

Fig. 7: Realistic Ruhr District Scenario (Section)

![Image](Image.png)

Table 8: Experimental Setup

| Source Nodes | A, B, C, D |
| Destination Node | E |
| New Vehicles per second | 3 (1 per Node) |
| Simulation Time | 3600 seconds |
| Dijkstra Update Interval | 600 seconds |
| Tempo limits | 135 km/h (freeways), 80 km/h (highways) |
| Max Speed for Vehicles | 135 km/h |
| Vehicular Density Limits |
| Highways | \( \alpha = 35, \beta = 40 \) [vehicles/km], \( \gamma = 50, \delta = 10 \) [km/h] |
| 4-Lane Freeway | \( \alpha = 40, \beta = 55 \) [vehicles/km], \( \gamma = 50, \delta = 10 \) [km/h] |
| 6-Lane Freeway | \( \alpha = 40, \beta = 55 \) [vehicles/km], \( \gamma = 50, \delta = 10 \) [km/h] |

The traffic state classification from section IV is used to estimate congestion situations on different road types. See table 8 for experimental setup. The here described very basic 4-sources (A, B, C, D) and 1-sink (E) scenario yet produces realistic traffic congestions (with Dijkstra-based routing) characteristic to the chosen section of the Ruhr District\(^1\).

In figure 8 highest vehicle densities on different road types are plotted against the simulation time for Dijkstra-based routing and BeeJamA routing. Figure 9 depicts the distribution of individual travel times for vehicles arriving at their destination at an arrival time \( t \).

The traffic scenario consists of two freeways (fig. 7). Edges \( a \) and \( b \) correspond to the German freeway A40 connecting two large cities Dortmund and Bochum. Edge \( c \) corresponds to freeway A45 connecting northern and southern parts of Dortmund, and intersecting with the A40 (both freeways change from 4-lanes to 6-lanes alternately). The remaining edges in the scenario are classified as highways (see table 8 for experimental setup).

In the Dijkstra-based routing experiment, the traffic generated at nodes A, B, C and D (University of Dortmund campus and a large industrial area) initially utilizes both freeway edges \( a \) and \( b \) to reach the common destination node E (corresponding to a large residential area in eastern Bochum). After 300 sec. local congestion clusters emerge with approx. 10-15 vehicles involved while the overall traffic remains fluent. After about 500 sec. vehicles on both edges \( a \) and \( b \) are piling up and congestions occur. Individual travel times increase considerably. In reaction to this, cars from source nodes A, C and D are routed to take edge \( d \) which is less populated at that moment and thus receives a better rating. Vehicles from node B continue to take edge \( b \) since cars originating here are mostly unaffected by the congestion tailback on the highways leading towards the freeway edge \( b \).

As a result, after 800 sec. vehicles with low travel times of approx. 220 sec. which were routed over the empty edge \( d \) reach their destinations. At the same time delayed vehicles routed over the jammed edge \( a \) arrive at node E with travel times of 500 sec. and higher (see fig. 9).

Traffic on the highway edge \( d \) quickly builds up and travel times increase. At the same time congestions on the freeway dissolve and travel times for vehicles from node B (routed over \( a \) and \( b \)) improve drastically. Thus, after 1200 sec. vehicles are again rerouted to take the now enhanced freeway edges \( a \) and \( b \). This oscillating behavior is observed throughout the remaining simulations and is depicted in figures 8 and 9.

With BeeJamA routing vehicles are prorated to take both freeways and highways at dynamic rates. At the beginning

\(^1\) Auxiliary note by one of the authors: The congestion scenario here described is observed by the authors on a daily basis on work days, during rush hours from 16:00 to 18:00.
of the simulation vehicles from nodes A, B, C and D are routed in the direction of freeway edge a as well as the diagonal highway d, directly connecting node A with the destination area (towards the latter at a smaller percentage). With traffic filling up and thus reducing the qualities of edges a and b highways are chosen more frequently. At approx. 600 sec. a somewhat stable yet fluent traffic situation is established in the scenario. This situation does not deteriorate throughout the remaining simulations.

In our experiments (after an initial simulation interval of approx. 200 sec.) traffic densities on all monitored roads are smaller with BeeJamA routing compared to corresponding road types in Dijkstra routing (see fig. 8). With Dijkstra-based routing heavy congestions occur after approx. 1800 sec. (densities of 55 vehicles per km and more). With BeeJamA routing the system remains congestion free and average travel times are lower (or equal at most) than corresponding average travel times with Dijkstra-based routing.

VI. CONCLUSION

The target area of our current work is the Ruhr District, the largest and very densely populated industrial region in Europe (compare section V). It is built from a conglomerate of more than 10 cities along the Ruhr River. While this structure poses a very high practical challenge for traffic regulation, its very dense and complex intra-city and inter-city road system is at the same time an unrivalled reservoir for Swarm Intelligence based approaches like BeeJamA. In order to solve the highly dynamic traffic congestion problem we have introduced, in the BeeJamA project, highly adaptive multi-layer routing algorithms which are meant to direct cars or trucks from road intersection to intersection. For this purpose we built upon our own work in adaptive network routing (BeeHive project). The major breakthrough stems from an extensive survey of traffic data allowing for empirically defining a mathematical quality rating function for local probability decision making.

Our algorithms are robust in the sense that not only would unforeseen events (accidents) create blockings (deadlocks) but even in case of drivers who do not follow the directions these do not really spoil the system: There are still advantages for drivers who follow the suggestions. This is a key point for introducing BeeJamA into practice, a consequence of the complete decentralization of all control actions.)

While the results just mentioned are conceptually not unexpected our extensive simulation experiments document a very substantial advantage of BeeJamA over traditional congestion handling.

REFERENCES